# CS585: Big Data Management

# Project 2

**<u>Total Points:</u>  150**

**<u>Release Date</u>:  02/16/2017**

**<u>Due Date:</u>  02/26/2017 (11:59PM)**

**<u>Teams:</u> Project to be done in teams of two.**

## Short Description

In this project, you will write java map-reduce jobs that implement advanced operations in Hadoop as well as learn more details about Hadoop's Input Formats.

## Problem 1 (Spatial Join) [50 points]

Spatial join is a common type of joins in many applications that manage multi-dimensional data. A typical example of spatial join is to have two datasets: ***Dataset P*** (set of points in two dimensional space) as shown in Figure 1a, and ***Dataset R*** (set of rectangles in two dimensional space) as shown in Figure 1b. The spatial join operation is to join these two datasets and report any pair (rectangle *r*, point *p*) where *p* is contained in *r* (or even in the border of *r*).

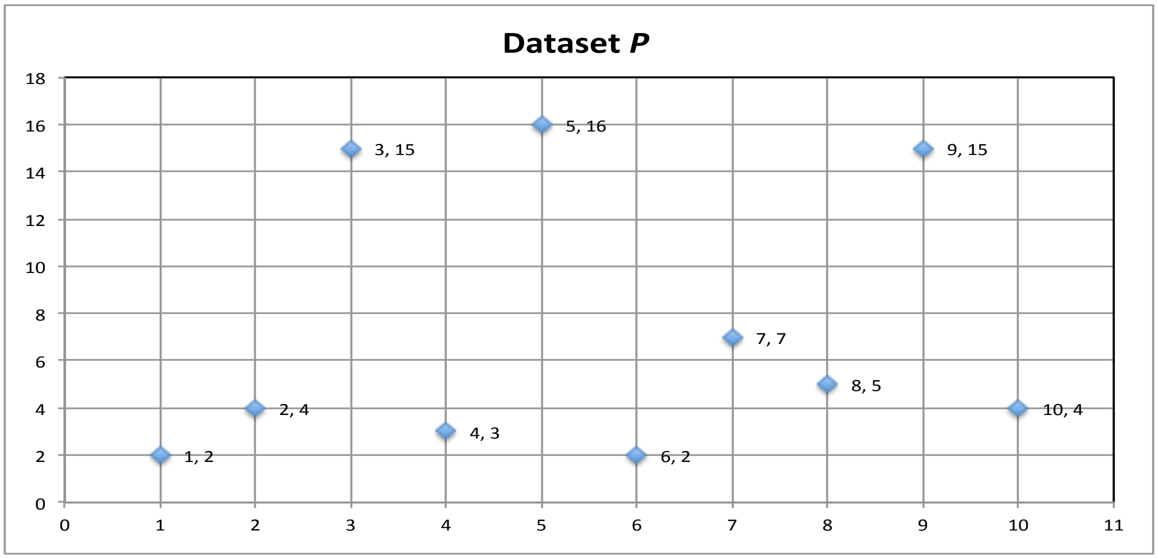


**Figure 1a: Set of 2D Points**

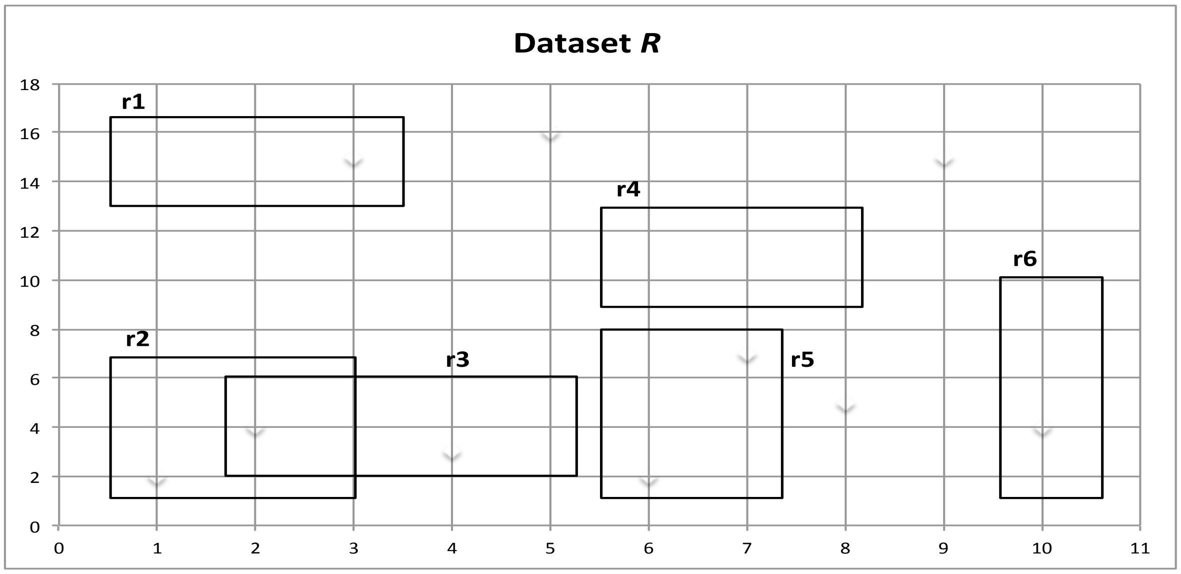


**Figure 1b: Set of 2D Rectangles**

For example, the join between the two datasets shown in Figure 1, will result in.

<r1, (3,15)>
<r2, (1,2)>
<r2, (2,4)>
<r3, (2,4)>
<r3, (4,3)>
<r5, (6,2)>
<r5, (7,7)>
<r6, (10,4)>

**Step 1 (Create the Datasets)[10 Points]**

- Your task in this step is to create the two datasets *P* (set of 2D points) and *R* (set of 2D rectangles). Assume the space extends from 1…10,000 in the both the X and Y axis. Each line will contain one object.
- Scale each dataset *P* or *R* to be at least 100MB.
- Choose the appropriate random function (of your choice) to create the points. For the rectangles, you will need to also select a point at random (say the top-left corner), and then select two random variables that define the *height* and *width* of the rectangle. For example, the height random variable can be uniform between [1,20] and the width is also uniform between [1,5].

**Step 2 (MapReduce job for Spatial Join)[40 Points]**

In this step, you will write a java map-reduce job that implements the spatial join operation between the two datasets *P* and *R* based on the following requirements:

- The program takes an optional input parameter W(x1, y1, x2, y2) that indicate a spatial window (rectangle) of interest within which we want to report the joined objects. If W is omitted, then the entire two sets should be joined.
    - Example, referring to Figure 1, if the window parameter is W(1, 3, 3, 20), then the reported joined objects should be:

      <r1, (3,15)>
      <r2, (2,4)>
      <r3, (2,4)>

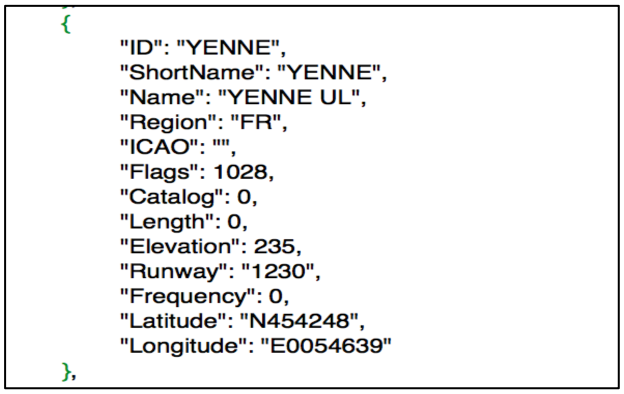
- You should have a single map-reduce job to implement the spatial join operation.

## Problem 2 (Custom Input Format) [50 points]

So far, all of the given assignments use text files as input, and hence you use 'TextInputFormat()' to read the files. In this problem, you will learn more about Hadoop input formats and you will write your custom one to read the input data.

### Step 1 (Data Sets)

You will use the dataset posted in Blackboard System (under Project 2), the file name is "airfield.text". This file has records formatted in JSON format. Each record starts with "{" and ends with "}" (no quotes). All attributes in between form one record. Records are separated with "," For example, the following image shows one record:



```
{
        "ID": "YENNE",
        "ShortName": "YENNE",
        "Name": "YENNE UL",
        "Region": "FR",
        "ICAO": "",
        "Flags": 1028,
        "Catalog": 0,
        "Length": 0,
        "Elevation": 235,
        "Runway": "1230",
        "Frequency": 0,
        "Latitude": "N454248",
        "Longitude": "E0054639"
},
```

Upload this file into HDSF.

### Step 2 (Map Job with a Custom Input Format)[50 Points]

- Now, to do any job on the above dataset using the standard "TextInputFormat()", the map function must be complex as it needs to collect many lines to form a single record. This complexity will repeat with each written job over the above dataset.
- A better way is to write a custom input format, call it "***JSONInputFormat***". This input format should read many lines from the input file until it gets a complete record (as the one in the image above), and then coverts these lines to a list of comma separated values in a single line, and then pass it to the map function.
  - E.g., each input to the map function should be: *ID:..., ShortName:...., Name:..., Region:..., ...*
  - In this case, the map function is not even aware that it is reading JSON formatted file.
  - As you see the line should have the fieldname then colon and then the value, and then "," separating the fields.

- Your task is to write this new "JSONInputFormat", and use it in a map-reduce job that aggregates the records based on the "Flag" field, and for each flag value report the number of corresponding records.

- Part of this step is to control the number of mappers that will execute to process the input file. We need to divide the file (independent of the HDFS block size) into 5 splits, which means Hadoop should start 5 mappers to process the file.

- ***Hint:*** You need to understand first the "***FileInputFormat***", "***TextInputFormat***", and "***LineRecordReader***" classes. And you can reuse some of them and build your new one as extension.

## Problem 3 (K-Means Clustering) [50 points]

K-Means clustering is a popular algorithm for clustering similar objects into *K* groups (clusters). It starts with an initial seed of K points (randomly chosen) as centers, and then the algorithm iteratively tries to enhance these centers. The algorithm terminates either when two consecutive iterations generate the same K centers, i.e., the centers did not change, or a maximum number of iterations is reached.

*Hint: You may reference these links to get some ideas (in addition to the course slides):*

http://en.wikipedia.org/wiki/K-means_clustering#Standard_algorithm
https://cwiki.apache.org/confluence/display/MAHOUT/K-Means+Clustering

### Step 1 (Creation of Dataset) [10 points]:

- Create a dataset that consists of 2-dimenional points, i.e., each point has (x, y) values. X and Y values each range from 0 to 10,000. Each point is in a separate line.
- Scale the dataset such that its size is around 100MB.
- Create another file that will contain K initial seed points. ***Make the "K" value as a parameter to your program***, such that your program will generate these K seeds randomly, and then you upload the generated file to HDFS.

### Step 2 (Clustering the Data) [40 points]:

Write map-reduce job(s) that implement the K-Means clustering algorithm as given in the course slides. The algorithm should terminates if either of these two conditions become true:

a) The K centers did not change over two consecutive iterations
b) The maximum number of iterations (make it five (5) iterations) has reached.

- Apply the tricks given in class and in the 2nd link above such as:
  - Use of a combiner
  - Use a single reducer
  - The reducer should indicate in its output file whether centers have changed or not.

Hint: Since the algorithm is iterative, then you need your program that generates the map-reduce jobs to control whether it should start another iteration or not.

## What to Submit

You will submit a single zip file containing the java code needed to answer the queries above. Also include a .doc or .pdf report file containing any required documentation.

## How to Submit

Use blackboard system to submit your files.